



# CNET\_LAB\_ASSIGNMENT#1

22i-1552\_Syed Arham Ahmed

First, I wrote code to send messages to the server until the client enters “end”. Until then the server prints whatever the client enters. The changes that I made in my code so that I could use another computer act as a client and send messages to the server is as below:

```

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <sys/types.h>
4 #include <sys/socket.h>
5 #include <unistd.h>
6 #include <netinet/in.h>
7 #include <cstring>
8 #include <arpa/inet.h>
9
10 int main()
11 {
12     char buf[200];
13
14     int server_socket;
15     server_socket = socket(AF_INET, SOCK_STREAM, 0);
16
17
18     struct sockaddr_in server_address;
19     server_address.sin_family = AF_INET;
20     server_address.sin_port = htons(3001);
21     server_address.sin_addr.s_addr = inet_addr("192.168.1.10");
22
23     bind(server_socket, (struct sockaddr*) &server_address, sizeof(server_address));
24     listen(server_socket, 5);
25
26     int client_socket;
27     client_socket = accept(server_socket, NULL, NULL);
28
29     while (1)
30     {
31         memset(buf, 0, sizeof(buf));
32         recv(client_socket, &buf, sizeof(buf), 0);
33
34         if (strcmp(buf, "end") == 0)
35         {
36             printf("Client has terminated the connection.\n");
37             break;
38         }
39         printf("\nClient: %s\n", buf);
40     }
41
42     close(server_socket);
43     return 0;
44 }

```

The above code is my server side code and In line 21 I have changed the code from “server\_address.sin\_addr.s\_addr = INADDR\_ANY;” to “server\_address.sin\_addr.s\_addr = inet\_addr("192.168.1.10");” in the brackets I have specified the IP address of my server which is a Kali VM. I also included the arpa/inet.h header file in the code.

That’s the only change I made in the server side, now let’s take a look at the client side.

```

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <sys/types.h>
4 #include <sys/socket.h>
5 #include <unistd.h>
6 #include <netinet/in.h>
7 #include <string.h>
8 #include <cstring>
9 #include <arpa/inet.h>
10
11 int main()
12 {
13     char request[256];
14     char buf[200];
15
16     int sock;
17     sock = socket(AF_INET, SOCK_STREAM, 0);
18
19     struct sockaddr_in server_address;
20     server_address.sin_family = AF_INET;
21     server_address.sin_addr.s_addr = inet_addr("192.168.1.10");
22     server_address.sin_port = htons(3001);
23     connect(sock, (struct sockaddr *) &server_address, sizeof(server_address));
24
25     while(1)
26     {
27         printf("Enter message (type 'end' to terminate): ");
28         scanf("%s", request);
29
30         send(sock, request, strlen(request), 0);
31         if (strcmp(request, "end")==0)
32         {
33             break;
34         }
35     }
36
37     close(sock);

```

The above code is my client side code that is being run on an Ubuntu vm on a separate computer. Again over here I have also specified the server IP in line 21, changed it from “server\_address.sin\_addr.s\_addr = INADDR\_ANY;” to the line you can see in the screenshot above.

Now these are the changes that I made to the code itself, but this was not it, I had to do further things so that my computers could communicate as currently they were un-able to.

First, I changed the network settings of my Kali VM and changed it from NAT to Bridged Adapter with the intel R Wi-Fi name as shown in the screenshot below. I did the same for the Ubuntu VM on my other computer. These changes enabled both of my VM’s to be on the same network, **as a bridged adapter acts as a switch between two computers.**



Now that I had configured everything, I pinged my server from the client side just to make sure they were able to connect.

```
arham@arham-VirtualBox:~/Desktop/CNET lab/Assignment # 1$ ping 192.168.1.10
PING 192.168.1.10 (192.168.1.10) 56(84) bytes of data.
 64 bytes from 192.168.1.10: icmp_seq=1 ttl=64 time=81.8 ms
 64 bytes from 192.168.1.10: icmp_seq=2 ttl=64 time=7.41 ms
 64 bytes from 192.168.1.10: icmp_seq=3 ttl=64 time=103 ms
^C
--- 192.168.1.10 ping statistics ---
 3 packets transmitted, 3 received, 0% packet loss, time 2001ms
 rtt min/avg/max/mdev = 7.414/64.056/102.930/40.968 ms
arham@arham-VirtualBox:~/Desktop/CNET lab/Assignment # 1$
```

Now that everything was working fine, I first compiled my server code and ran it ie: KALI VM, then on my other computer I compiled my client code and ran it ie: Ubuntu VM, I then gave input to server and server side displayed my input which shows that I achieved my goal.

#### Client-Side Terminal:

```
arham@arham-VirtualBox:~/Desktop/CNET lab/Assignment # 1$ ./latestclient
Enter message (type 'end' to terminate): hello
Enter message (type 'end' to terminate): 1552
Enter message (type 'end' to terminate): arham
Enter message (type 'end' to terminate): bs-cy-b
Enter message (type 'end' to terminate): SyedArhamAhmed_22i-1552_CNET_Assignment#1
Enter message (type 'end' to terminate): end
arham@arham-VirtualBox:~/Desktop/CNET lab/Assignment # 1$
```

#### Server-Side Terminal:

```
(arham@kali)-[~/Desktop/CNET Lab/Lab 4]
$ ./server
Client: hello
Client: 1552
Client: arham
Client: bs-cy-b
Client: SyedArhamAhmed_22i-1552_CNET_Assignment#1
Client has terminated the connection.
```

## 22i-1552\_BS-CY-B\_Syed Arham Ahmed\_CNET\_LAB\_Assignment#1

Now that we have seen that our client server connection is working properly, we will use Wireshark to capture the packets and see what messages were exchanged.

No.	Time	Source	Destination	Protocol	Length	Info
20	6.65874831	192.168.1.9	192.168.1.10	TCP	74	38836 → 3801 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=1588513462 TSecr=0 WS=128
21	6.655172631	192.168.1.10	192.168.1.9	TCP	74	3801 → 38836 [SYN, ACK] Seq=8 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=3281691623 TSecr=1588513462 WS=1024
22	6.641554842	192.168.1.9	192.168.1.10	TCP	66	38836 → 3801 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=1588513513 TSecr=3281691623
23	6.676910683	192.168.1.9	192.168.1.10	TCP	71	38836 → 3801 [PSH, ACK] Seq=1 Ack=1 Win=64256 Len=5 TSval=1588515130 TSecr=3281691623
24	6.676977369	192.168.1.10	192.168.1.9	TCP	66	3801 → 38836 [ACK] Seq=1 Ack=5 Win=65336 Len=0 TSval=3281693265 TSecr=1588515139
25	21.206442212	192.168.1.9	192.168.1.10	TCP	97	38836 → 3801 [PSH, ACK] Seq=6 Ack=1 Win=64256 Len=31 TSval=1588526622 TSecr=3281693265
26	20.2096502254	192.168.1.10	192.168.1.9	TCP	66	3801 → 38836 [ACK] Seq=1 Ack=37 Win=65536 Len=0 TSval=3281706788 TSecr=1588526622
27	23.641855160	192.168.1.9	192.168.1.10	TCP	69	38836 → 3801 [PSH, ACK] Seq=37 Ack=1 Win=64256 Len=3 TSval=1588530460 TSecr=3281706788
28	23.641914738	192.168.1.9	192.168.1.10	TCP	66	3801 → 38836 [ACK] Seq=1 Ack=40 Win=65536 Len=0 TSval=3281706629 TSecr=1588530460
29	32.6509375502	192.168.1.10	192.168.1.9	TCP	69	38836 → 3801 [PSH, ACK] Seq=40 Ack=1 Win=64256 Len=8 TSval=1588532546 TSecr=3281706629
30	35.6994433975	192.168.1.10	192.168.1.9	TCP	66	3801 → 38836 [ACK] Seq=1 Ack=43 Win=65536 Len=0 TSval=3281710678 TSecr=1588532546
31	35.6994379728	192.168.1.9	192.168.1.10	TCP	66	38836 → 3801 [FIN, ACK] Seq=43 Ack=43 Win=64256 Len=0 TSval=1588532546 TSecr=3281706629
32	35.699825886	192.168.1.10	192.168.1.9	TCP	66	3801 → 38836 [FIN, ACK] Seq=1 Ack=44 Win=65536 Len=0 TSval=3281710678 TSecr=1588532546
36	25.121907625	192.168.1.9	192.168.1.10	TCP	66	38836 → 3801 [ACK] Seq=44 Ack=2 Win=64256 Len=0 TSval=1588532582 TSecr=3281710678

As we can see in the screenshot that I have filtered out the packets that were sent and received from my server to the client.

Now I will open the packets and see my messages.

[illegible]

As we can see in the above screenshot, these are the messages that my client computer sent to my server computer.

```
(arham@kali)-[~/Desktop/CNET_Lab/Lab_4]
$ ./server
Client: hello
Client: 22i-1552_BS-CY-B_SyedArhamAhmed
Client: bye
Client has terminated the connection.
```

## 22i-1552\_BS-CY-B\_Syed Arham Ahmed\_CNET\_LAB\_Assignment#1

Now let's answer the questions asked in the assignment manual.

### Q.1. What is the source IP address?

The source IP address is the client as it initiates the connection, so it is 192.168.1.9

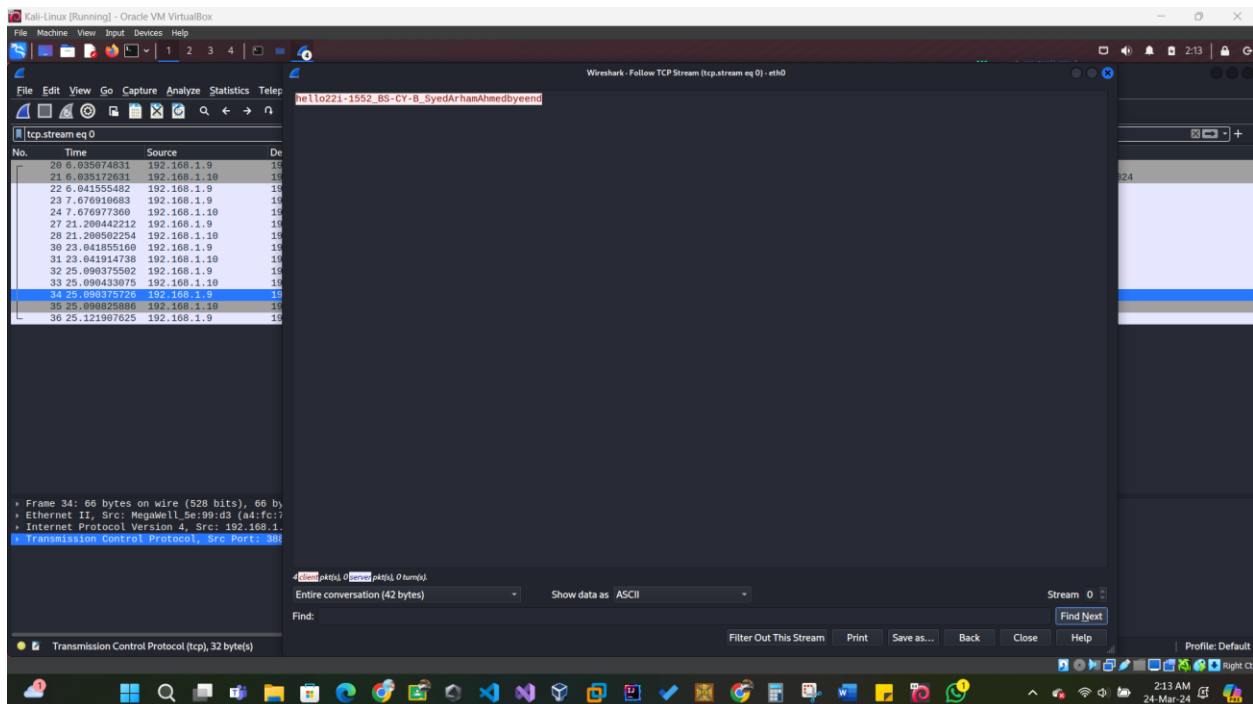
### Q.2. What is the destination IP address?

The destination IP address is the server which received the client's message, so it is 192.168.1.10

### Q.3. Using Wireshark, find out what message was sent by client to server.

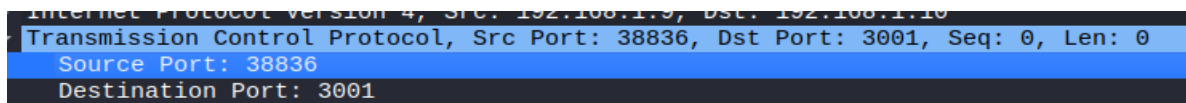
Already answered it above, just attaching the screenshot for reference. The messages were:

Hello,22i-1552\_BS-CY-B\_SyedArhamAhmed,bye,end



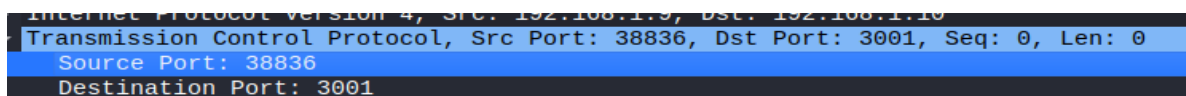
### Q.4. What is the source port number?

As I mentioned above that client is the source, so its port number is 38836.



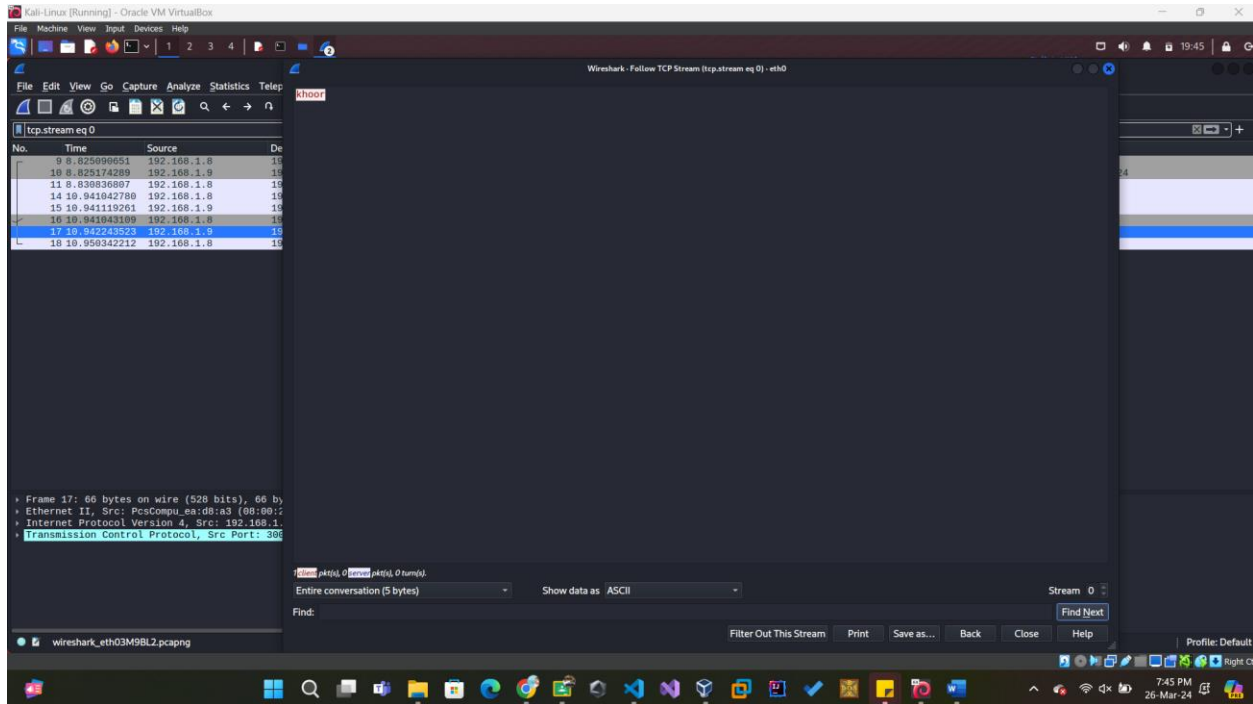
### Q.5. What is the destination port number?

The destination is the server, and its port number is 3001.



Now I will modify my code so that Wireshark can't just tell me the message directly. For that I can just simply encrypt the message before sending it and decrypt it on the server side.

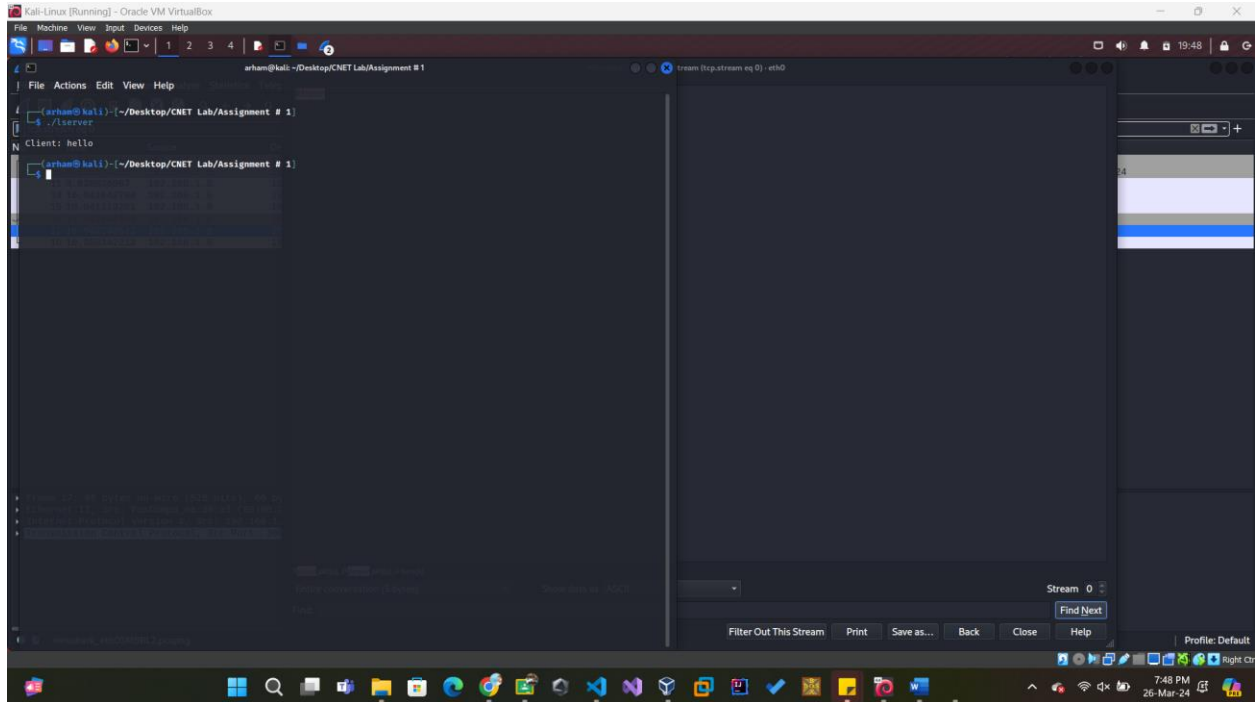
**Q.6. After modifying code, show what difference you have observed in Wireshark after step no 11.**



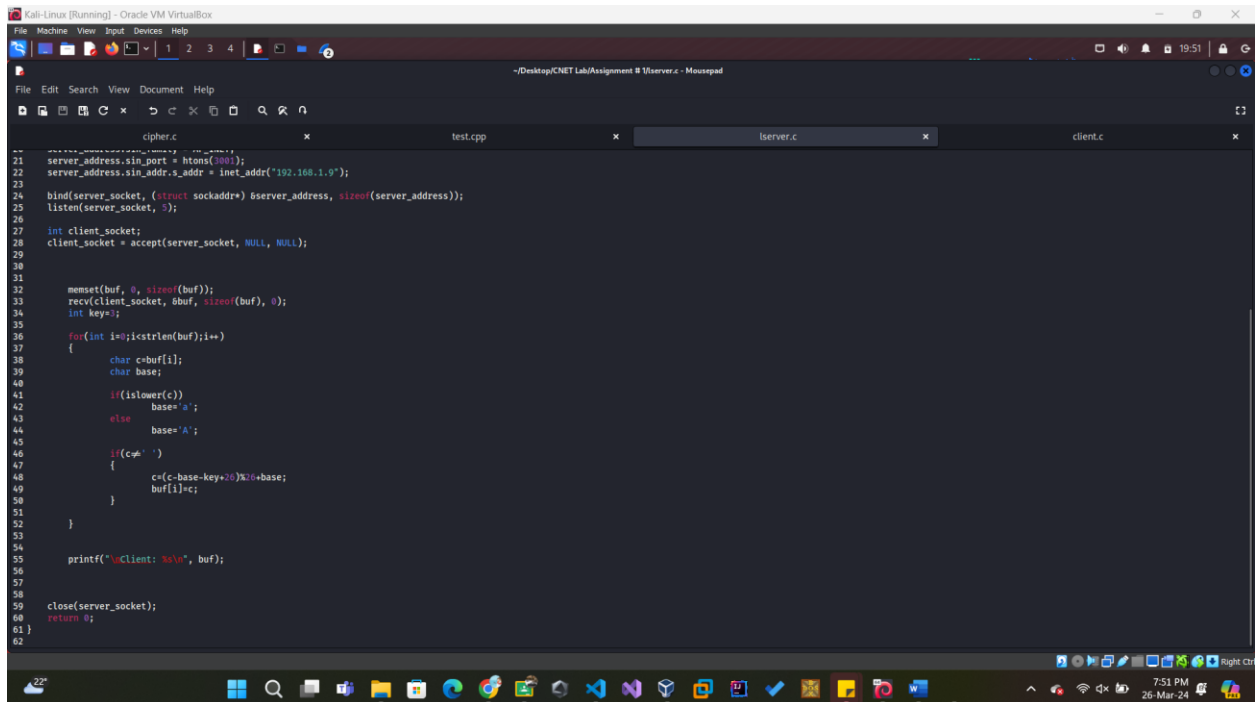
As now I have applied encryption to my message at the client i.e. senders end, Wireshark shows me the encrypted word and the packet sniffer cannot make any sense out of it.



## 22i-1552\_BS-CY-B\_Syed Arham Ahmed\_CNET\_LAB\_Assignment#1

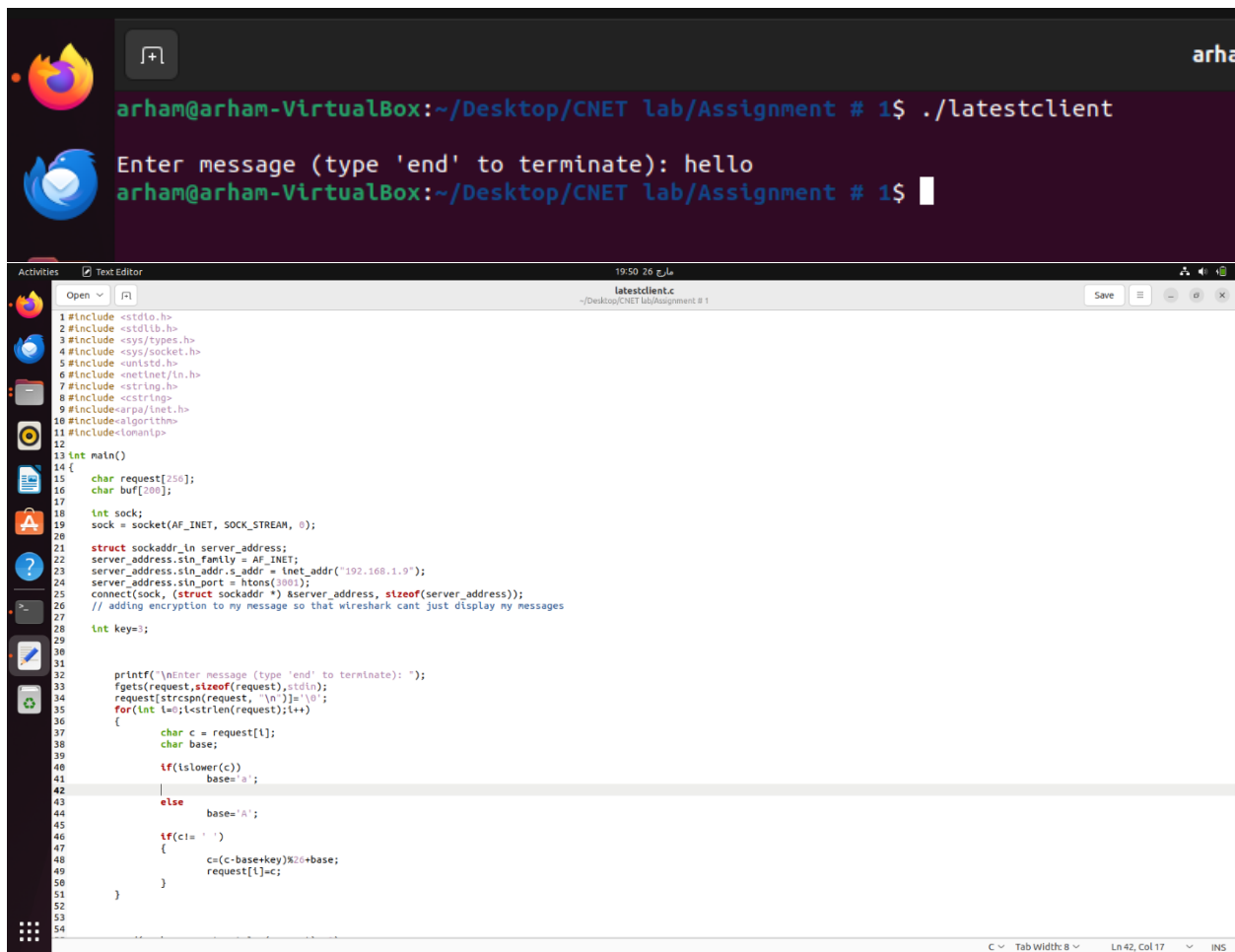


But on the receiver side the message hello is received correctly as I have applied decryption at the server end.



The above code is my server code which is decrypting the received message. I have used Caesar cipher here.





The screenshot shows a Linux desktop environment. At the top, a terminal window displays the command `./latestclient` being executed in the directory `~/Desktop/CNET lab/Assignment # 1`. The prompt shows the user is `arham` on a machine named `arham-VirtualBox`. The terminal output shows the prompt `Enter message (type 'end' to terminate):` followed by the user input `hello`. Below the terminal, a text editor window titled `latestclient.c` shows the source code of the client program. The code includes standard headers, sets up a socket, connects to a server at `192.168.1.9` on port `3001`, and implements a Caesar cipher encryption function. The encryption function takes a message and a key, and shifts each character by the key value. The code is as follows:

```

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <sys/types.h>
4 #include <sys/socket.h>
5 #include <unistd.h>
6 #include <netinet/in.h>
7 #include <string.h>
8 #include <string>
9 #include <arpa/inet.h>
10 #include <algorithm>
11 #include <unistd.h>
12
13 int main()
14 {
15     char request[256];
16     char buf[200];
17
18     int sock;
19     sock = socket(AF_INET, SOCK_STREAM, 0);
20
21     struct sockaddr_in server_address;
22     server_address.sin_family = AF_INET;
23     server_address.sin_addr.s_addr = inet_addr("192.168.1.9");
24     server_address.sin_port = htons(3001);
25     connect(sock, (struct sockaddr *) &server_address, sizeof(server_address));
26     // adding encryption to my message so that wireshark cant just display my messages
27
28     int key=1;
29
30
31     printf("\nEnter message (type 'end' to terminate): ");
32     fgets(request, sizeof(request), stdin);
33     request[strcspn(request, "\n")] = '\0';
34     for(int i=0; i<strlen(request); i++)
35     {
36         char c = request[i];
37         char base;
38         if(islower(c))
39             base = 'a';
40         else
41             base = 'A';
42         if(c != ' ')
43         {
44             c = (c - base + key) % 26 + base;
45             request[i] = c;
46         }
47     }
48     printf("\nEncrypted message: %s", request);
49 }

```

The above is my client code i.e. sender code and I have encrypted my message here using Caesar cipher. I entered “hello” which was encrypted and become “khor” as we can see in the Wireshark packet, but on the receiver side we can easily understand and view the message as it was decrypted after being received.

That’s all for the report, thank you for reading till the end!